项目文档

# Functional Requirement

1. Functional Requirements  
  
1.1 Customer Registration Function   
 Function ID: FR-01   
 Description: Customers can create a new account by providing personal information. The system validates the email for uniqueness and the password for strength.   
 Input: Customer's personal information including name, email, and password.   
 Output: A successfully registered customer account in the Customer data entity and a confirmation message sent to the customer.  
  
1.2 Customer Login Function   
 Function ID: FR-02   
 Description: Registered customers can log in using their email and password. The system verifies the credentials and logs the login activity.   
 Input: Customer's email and password.   
 Output: A logged-in customer session and a homepage or dashboard displayed to the customer.  
  
1.3 Customer Logout Function   
 Function ID: FR-03   
 Description: Customers can log out of their session. The system invalidates the session and logs the logout activity.   
 Input: Customer's request to log out.   
 Output: A logged-out customer session and the customer redirected to the login or home page.  
  
1.4 Product Browsing Function   
 Function ID: FR-04   
 Description: Customers can browse products by category and view product details.   
 Input: Customer's selected category or request to view product details.   
 Output: A list of products in the selected category or detailed product information displayed to the customer.  
  
1.5 Product Search Function   
 Function ID: FR-05   
 Description: Customers can search for products using keywords or filters.   
 Input: Search term or selected filters (e.g., category, price range).   
 Output: A list of search results including product name, price, and image.  
  
1.6 Shopping Cart Addition Function   
 Function ID: FR-06   
 Description: Customers can add products to their shopping cart. The system verifies inventory availability and updates the cart.   
 Input: Product ID and quantity selected by the customer.   
 Output: Updated shopping cart with the new item and confirmation message.  
  
1.7 Shopping Cart Modification Function   
 Function ID: FR-07   
 Description: Customers can modify the quantity or remove items from their shopping cart. The system recalculates the cart total.   
 Input: Cart item modification request (e.g., quantity change, item removal).   
 Output: Updated shopping cart and confirmation message.  
  
1.8 Shopping Cart Deletion Function   
 Function ID: FR-08   
 Description: Customers can clear all items from their shopping cart.   
 Input: Customer's request to delete the cart.   
 Output: An empty shopping cart and confirmation message.  
  
1.9 Order Placement Function   
 Function ID: FR-09   
 Description: Customers can place an order by confirming the cart and completing the checkout process. The system updates the inventory and sends a confirmation.   
 Input: Shopping cart with valid items and payment method selected by the customer.   
 Output: A new order record and inventory updated with reduced stock levels.  
  
1.10 Order Confirmation Function   
 Function ID: FR-10   
 Description: The system confirms the order and sends a confirmation message to the customer.   
 Input: Order details and customer contact information.   
 Output: Order confirmation status updated and a confirmation message sent to the customer.  
  
1.11 Order History Viewing Function   
 Function ID: FR-11   
 Description: Registered customers can view their previous orders and access order details.   
 Input: Request from the customer to view their order history.   
 Output: A list of past orders with details displayed to the customer.  
  
1.12 Inventory Management Function   
 Function ID: FR-12   
 Description: Administrators can manage product inventory levels, including updates and alerts for restocking.   
 Input: Product inventory update request from the administrator.   
 Output: Updated inventory levels and restocking alerts generated as needed.  
  
1.13 Product Category Creation Function   
 Function ID: FR-13   
 Description: Administrators can create new product categories.   
 Input: Category name, description, and other relevant details provided by the administrator.   
 Output: A new category added to the Product Category data entity.  
  
1.14 Product Category Update Function   
 Function ID: FR-14   
 Description: Administrators can update existing product categories.   
 Input: Category ID and updated details (e.g., name, description).   
 Output: Updated category information in the Product Category data entity.  
  
1.15 Product Category Deletion Function   
 Function ID: FR-15   
 Description: Administrators can delete product categories after reassigning or removing associated products.   
 Input: Category ID and confirmation to delete.   
 Output: Deleted category and updated product records.  
  
1.16 Payment Processing Function   
 Function ID: FR-16   
 Description: Customers can process payments during checkout. The system verifies the payment and updates the order and inventory.   
 Input: Payment method and payment details provided by the customer.   
 Output: Confirmed payment status and updated order and inventory data.  
  
1.17 Payment Information Update Function   
 Function ID: FR-17   
 Description: Customers can update their stored payment information.   
 Input: Updated payment method details provided by the customer.   
 Output: Updated payment information in the Payment data entity.  
  
1.18 Contact Information Management Function   
 Function ID: FR-18   
 Description: Customers can add, modify, or delete their contact information.   
 Input: Updated or new contact details (e.g., address, phone number).   
 Output: Updated contact information in the Contact Information data entity.  
  
1.19 Plugin Installation Function   
 Function ID: FR-19   
 Description: Administrators can install plugins to extend system functionality.   
 Input: Plugin name and installation request from the administrator.   
 Output: Installed and activated plugin, with event logged in the Plugin data entity.  
  
1.20 Plugin Configuration Function   
 Function ID: FR-20   
 Description: Administrators can configure the settings of an installed plugin.   
 Input: Plugin configuration settings provided by the administrator.   
 Output: Updated plugin configuration in the PluginConfiguration data entity.  
  
1.21 Plugin Uninstallation Function   
 Function ID: FR-21   
 Description: Administrators can uninstall plugins after ensuring they are not in use.   
 Input: Plugin ID and uninstall request from the administrator.   
 Output: Removed plugin and associated configurations, with event logged.  
  
1.22 Manage Order Confirmation Function   
 Function ID: FR-22   
 Description: Administrators can modify the confirmation status of an order and send updated notifications to customers.   
 Input: Order confirmation status change request from the administrator.   
 Output: Updated order confirmation status and notification sent to the customer.  
  
1.23 Manage Cart Item Function   
 Function ID: FR-23   
 Description: Customers can modify or remove specific items in their shopping cart. The system updates the cart and inventory.   
 Input: Cart item modification or deletion request from the customer.   
 Output: Updated shopping cart and inventory levels if necessary.  
  
1.24 Manage Plugin Configuration Function   
 Function ID: FR-24   
 Description: Administrators can modify the configuration settings of an installed plugin.   
 Input: Plugin configuration update request from the administrator.   
 Output: Updated plugin configuration in the PluginConfiguration data entity.  
  
1.25 Manage Product Function   
 Function ID: FR-25   
 Description: Administrators can add, modify, or delete products and their associated category and inventory.   
 Input: Product details (e.g., name, price, category, inventory) from the administrator.   
 Output: Updated product records in the Product, Product Category, and Inventory data entities.  
  
1.26 Manage Order Function   
 Function ID: FR-26   
 Description: Administrators can modify or process orders, including updating status and initiating refunds.   
 Input: Order modification or processing request from the administrator.   
 Output: Updated order status and inventory levels if necessary.  
  
1.27 Manage Administrator Function   
 Function ID: FR-27   
 Description: Super Administrators can add, modify, or deactivate other administrators.   
 Input: Administrator details (e.g., username, role, status) from the super administrator.   
 Output: Updated administrator records in the Administrator data entity.  
  
1.28 Manage Payment Function   
 Function ID: FR-28   
 Description: Administrators can review and modify payment records, including initiating refunds.   
 Input: Payment modification or refund request from the administrator.   
 Output: Updated payment status and associated order records.  
  
1.29 Manage Plugin Function   
 Function ID: FR-29   
 Description: Administrators can activate, deactivate, or remove plugins and their configurations.   
 Input: Plugin management request from the administrator.   
 Output: Updated plugin status and configurations in the Plugin and PluginConfiguration data entities.  
  
1.30 Manage Order Confirmation Function   
 Function ID: FR-30   
 Description: Administrators can update the confirmation status of an order and send notifications to the customer.   
 Input: Order confirmation status update request from the administrator.   
 Output: Updated order confirmation status and customer notification if required.

# External Description

2. External Interfaces   
The system interacts with various external interfaces to fulfill its functional requirements. These interfaces include user interfaces for customer and administrator interactions, software interfaces for data management, and communication interfaces for notifications and updates.   
  
2.1 User Interface Output   
The system provides a graphical user interface (GUI) for both customers and administrators. The GUI allows for intuitive interaction with the system through web-based and mobile platforms.   
  
- \*\*Customer Account Interface\*\*:   
 - \*\*Description\*\*: A web-based or mobile interface that enables customers to register, log in, and manage their accounts.   
 - \*\*Interaction Method\*\*: Form-based input for registration and login, with confirmation messages displayed on the screen.   
  
- \*\*Product Browsing Interface\*\*:   
 - \*\*Description\*\*: A user interface that allows customers to browse products by category and view product details.   
 - \*\*Interaction Method\*\*: Dropdown menus or clickable categories for navigation, product detail pages with images, descriptions, and pricing.   
  
- \*\*Product Search Interface\*\*:   
 - \*\*Description\*\*: A search interface that allows customers to input keywords or select filters to find products.   
 - \*\*Interaction Method\*\*: Search bar for keyword entry, filter options for refining results, and a results page displaying product name, price, and image.   
  
- \*\*Shopping Cart Interface\*\*:   
 - \*\*Description\*\*: A user interface for managing the shopping cart, including adding, modifying, and removing items.   
 - \*\*Interaction Method\*\*: Buttons or input fields to adjust quantities and remove items, with real-time updates to the cart total and confirmation messages.   
  
- \*\*Checkout Interface\*\*:   
 - \*\*Description\*\*: A user interface where customers confirm their orders and process payments.   
 - \*\*Interaction Method\*\*: Form-based input for payment method selection, with real-time validation and confirmation messages.   
  
- \*\*Order History Interface\*\*:   
 - \*\*Description\*\*: A user interface for customers to view their past orders and associated details.   
 - \*\*Interaction Method\*\*: List view of previous orders with clickable entries to expand and display order details.   
  
- \*\*Administrator Interface\*\*:   
 - \*\*Description\*\*: A web-based interface for administrators to manage inventory, categories, plugins, orders, and other administrative functions.   
 - \*\*Interaction Method\*\*: Dashboard-style interface with tabs for different administrative functions, form-based input for updates, and confirmation messages.   
  
2.2 Hardware Interface Output   
The system does not directly interact with any external hardware devices and does not require specific hardware interfaces.   
  
2.3 Software Interface Output   
The system interacts with several internal and external software systems and data entities to manage customer accounts, products, orders, and plugins.   
  
- \*\*Customer Data Entity\*\*:   
 - \*\*Description\*\*: Stores customer account information such as name, email, and password.   
 - \*\*Interaction Method\*\*: Data is added, modified, or retrieved through database queries initiated by the system during registration, login, and account management functions.   
  
- \*\*Product Data Entity\*\*:   
 - \*\*Description\*\*: Contains product details such as name, price, category, and inventory status.   
 - \*\*Interaction Method\*\*: Data is retrieved and displayed to customers during product browsing and search functions. Administrators can add, modify, or delete product records through the administrator interface.   
  
- \*\*Product Category Data Entity\*\*:   
 - \*\*Description\*\*: Stores information about product categories, including names and descriptions.   
 - \*\*Interaction Method\*\*: Categories are retrieved and displayed to customers during product browsing. Administrators can manage categories through the administrator interface.   
  
- \*\*Inventory Data Entity\*\*:   
 - \*\*Description\*\*: Tracks the stock levels of products.   
 - \*\*Interaction Method\*\*: Inventory data is updated automatically when customers add, modify, or remove items from their cart and when orders are placed or modified.   
  
- \*\*Shopping Cart Data Entity\*\*:   
 - \*\*Description\*\*: Stores the items selected by the customer for purchase, including product ID, quantity, and total price.   
 - \*\*Interaction Method\*\*: The cart is updated when customers add, modify, or remove items, and is cleared when the customer chooses to delete the cart.   
  
- \*\*Order Data Entity\*\*:   
 - \*\*Description\*\*: Stores order records, including customer information, product details, payment status, and order confirmation status.   
 - \*\*Interaction Method\*\*: Orders are created when a customer completes checkout. Administrators can update order status and initiate refunds.   
  
- \*\*Payment Data Entity\*\*:   
 - \*\*Description\*\*: Stores customer payment methods and payment records.   
 - \*\*Interaction Method\*\*: Payment data is updated when a customer places an order or modifies stored payment information. Administrators can review and modify payment records.   
  
- \*\*Contact Information Data Entity\*\*:   
 - \*\*Description\*\*: Stores customer contact details such as address and phone number.   
 - \*\*Interaction Method\*\*: Contact information is updated when a customer modifies or adds new contact details.   
  
- \*\*Plugin Data Entity\*\*:   
 - \*\*Description\*\*: Stores plugin information, including the plugin name, status, and installation timestamp.   
 - \*\*Interaction Method\*\*: Plugins are installed, activated, or uninstalled through the administrator interface, and the system logs these events.   
  
- \*\*PluginConfiguration Data Entity\*\*:   
 - \*\*Description\*\*: Stores configuration settings for installed plugins.   
 - \*\*Interaction Method\*\*: Administrators can configure or update plugin settings through the administrator interface, and the system updates the configuration accordingly.   
  
2.4 Communication Interface Output   
The system communicates with external services and users through various communication channels.   
  
- \*\*Email Notification Service\*\*:   
 - \*\*Description\*\*: Sends confirmation messages to customers after account registration, order placement, and payment processing.   
 - \*\*Interaction Method\*\*: The system sends an email to the customer's registered email address, including a confirmation message and order details.   
  
- \*\*Web Browsing Communication\*\*:   
 - \*\*Description\*\*: The system is accessed by customers and administrators through web browsers.   
 - \*\*Interaction Method\*\*: Customers and administrators interact with the system via HTTP/HTTPS protocols, with dynamic content loaded based on user actions.   
  
- \*\*Payment Gateway Integration\*\*:   
 - \*\*Description\*\*: Integrates with a third-party payment gateway to process customer payments.   
 - \*\*Interaction Method\*\*: The system sends payment method and transaction details to the payment gateway via API calls and receives a confirmation of payment status.   
  
- \*\*Restocking Alert Communication\*\*:   
 - \*\*Description\*\*: Sends alerts to administrators when inventory levels fall below a threshold.   
 - \*\*Interaction Method\*\*: Alerts are generated and sent via internal messaging or email when inventory levels require restocking.   
  
- \*\*Customer Order Status Notification\*\*:   
 - \*\*Description\*\*: Sends updates to customers regarding their order status.   
 - \*\*Interaction Method\*\*: Notifications are sent via email or in-system messages when an order is placed, confirmed, or modified by an administrator.   
  
All communication interfaces are designed to ensure secure and reliable interactions, with appropriate error handling and logging mechanisms in place.

# Use Case

Use Case Name: Customer Registration   
Use Case ID: UC-01   
Actors: Customer, System   
Preconditions:   
1. The customer has accessed the system's registration page.   
2. The system is operational and accessible.   
  
Postconditions:   
1. The customer is successfully registered in the system.   
2. A confirmation message is sent to the customer.   
3. The customer's information is stored in the Customer data entity.   
  
Main Flow:   
1. The customer provides personal information (e.g., name, email, password).   
2. The system validates the customer's email to ensure it is unique.   
3. The system checks the strength of the password provided.   
4. The system stores the customer's information in the Customer data entity.   
5. The system sends a confirmation message (e.g., via email) to the customer.   
6. The customer receives the confirmation message and completes registration.   
  
Alternative Flow:   
1. If the email provided is already registered, the system displays an error message and prompts the customer to enter a different email.   
2. If the password does not meet the strength requirements, the system informs the customer and asks for a stronger password.   
3. If the system fails to send the confirmation message, it logs an error and allows the customer to request a resend.  
  
Use Case Name: Customer Login   
Use Case ID: UC-02   
Actors: Customer, System   
Preconditions:   
1. The customer has already registered in the system and has a valid account.   
2. The customer has accessed the login page of the system.   
3. The system is operational and accessible.   
  
Postconditions:   
1. The customer is successfully authenticated and logged into the system.   
2. The system displays the customer's homepage or dashboard.   
3. The login activity is logged in the system for audit purposes.   
  
Main Flow:   
1. The customer enters their registered email and password on the login page.   
2. The system verifies the email and password against the Customer data entity.   
3. If the credentials are valid, the system logs the customer in and redirects them to their homepage.   
4. The system displays a welcome message and updates the session status to "logged in."   
  
Alternative Flow:   
1. If the email is not found in the Customer data entity, the system displays an error message and prompts the customer to check their email.   
2. If the password is incorrect, the system informs the customer and allows them to re-enter the password.   
3. If the customer exceeds the maximum number of failed login attempts, the system temporarily locks the account and notifies the customer.   
4. If the system encounters an unexpected error during authentication, it logs the error and displays a generic error message to the customer.  
  
Use Case Name: Customer Logout   
Use Case ID: UC-03   
Actors: Customer, System   
Preconditions:   
1. The customer is currently logged into the system.   
2. The system is operational and accessible.   
  
Postconditions:   
1. The customer is successfully logged out.   
2. The session status is updated to "logged out."   
3. The customer is redirected to the login or home page.   
  
Main Flow:   
1. The customer navigates to the logout option (e.g., through a dropdown menu or button).   
2. The system confirms the logout request.   
3. The system invalidates the current session and clears session data.   
4. The system redirects the customer to the login or home page.   
5. The system logs the logout activity for audit purposes.   
  
Alternative Flow:   
1. If the system fails to invalidate the session, it displays an error message and prompts the customer to try again.   
2. If the customer closes the browser or navigates away before confirmation, the system automatically logs out after a timeout period.   
3. If the system is unable to redirect the customer, it displays a message confirming logout and provides a link to the home or login page.  
  
Use Case Name: Product Browsing   
Use Case ID: UC-04   
Actors: Customer, System   
Preconditions:   
1. The customer has accessed the product catalog page.   
2. The system is operational and accessible.   
3. The product data is available in the Product data entity.   
  
Postconditions:   
1. The customer can view product details.   
2. The system displays the relevant products based on search or category.   
3. The browsing activity is recorded in the system for analytics purposes.   
  
Main Flow:   
1. The customer selects a product category from the Product Category data entity.   
2. The system retrieves and displays the list of products in the selected category from the Product data entity.   
3. The customer can search for specific products using keywords.   
4. The system filters and displays the matching products.   
5. The customer views product details such as name, price, description, and image.   
6. The customer can add a product to their Shopping Cart for later purchase.   
  
Alternative Flow:   
1. If no products are found in the selected category, the system displays a message indicating the absence of products.   
2. If the search returns no results, the system informs the customer and suggests alternative search terms.   
3. If the system cannot retrieve product data, it logs an error and displays a message to the customer.   
4. If the customer attempts to view a product that is out of stock, the system displays a message indicating the inventory status and suggests similar products.  
  
Use Case Name: Product Search   
Use Case ID: UC-05   
Actors: Customer, System   
Preconditions:   
1. The customer is logged into the system or has accessed the product catalog.   
2. The system is operational and accessible.   
3. The Product data entity contains the necessary product information.   
  
Postconditions:   
1. The customer receives a list of products matching their search criteria.   
2. The system displays the relevant product details.   
3. The search activity is recorded for analytics or personalization purposes.   
  
Main Flow:   
1. The customer enters a search term or selects filters (e.g., category, price range) on the search interface.   
2. The system processes the search query and retrieves matching products from the Product data entity.   
3. The system displays the list of search results, including product name, price, and image.   
4. The customer can refine the search further by applying additional filters or sorting options.   
5. The system updates the displayed results based on the refined criteria.   
6. The customer selects a product to view more details or adds it to the Shopping Cart.   
  
Alternative Flow:   
1. If the search returns no results, the system displays a message and suggests alternative search terms.   
2. If the search query is invalid or incomplete, the system prompts the customer to correct the input.   
3. If the system cannot access the Product data entity, it logs an error and displays a message to the customer.   
4. If a product is out of stock, the system displays the inventory status and may suggest similar products.  
  
Use Case Name: Shopping Cart Addition   
Use Case ID: UC-06   
Actors: Customer, System   
Preconditions:   
1. The customer is logged into the system or is browsing as a guest.   
2. The system is operational and accessible.   
3. The product to be added is available in the Product data entity.   
  
Postconditions:   
1. The selected product is successfully added to the Shopping Cart.   
2. The Shopping Cart data entity is updated to reflect the new item.   
3. The customer receives a confirmation that the product was added.   
  
Main Flow:   
1. The customer selects a product from the Product catalog or search results.   
2. The system displays the product details.   
3. The customer chooses the quantity and clicks the "Add to Cart" button.   
4. The system verifies the product's availability in the Inventory data entity.   
5. The system adds the product and quantity to the Shopping Cart data entity.   
6. The system updates the Shopping Cart summary (e.g., item count, total price).   
7. The system provides a confirmation message (e.g., "Item added to cart").   
8. The customer can choose to continue shopping or proceed to checkout.   
  
Alternative Flow:   
1. If the product is out of stock, the system displays an error message and suggests similar products.   
2. If the quantity exceeds available inventory, the system adjusts the quantity to the maximum available and informs the customer.   
3. If the system cannot update the Shopping Cart, it logs an error and displays a message to the customer.   
4. If the customer is not logged in and adds an item to the cart, the system stores the cart in a session and prompts the customer to log in to save it permanently.  
  
Use Case Name: Shopping Cart Modification   
Use Case ID: UC-07   
Actors: Customer, System   
Preconditions:   
1. The customer is logged into the system or is browsing as a guest with a session-based Shopping Cart.   
2. The Shopping Cart contains at least one item.   
3. The system is operational and accessible.   
  
Postconditions:   
1. The Shopping Cart is updated based on the customer's modifications (e.g., quantity change, item removal).   
2. The Shopping Cart data entity is updated accordingly.   
3. The customer receives a confirmation of the modification.   
  
Main Flow:   
1. The customer accesses the Shopping Cart page.   
2. The system displays the current items in the Shopping Cart.   
3. The customer selects an item to modify (e.g., change quantity, remove item).   
4. The system updates the Shopping Cart data entity based on the modification.   
5. The system recalculates the total price and updates the cart summary.   
6. The system provides a confirmation message (e.g., "Cart updated successfully").   
7. The customer can choose to continue shopping or proceed to checkout.   
  
Alternative Flow:   
1. If the customer attempts to set a quantity higher than the available inventory, the system adjusts the quantity to the maximum available and displays a warning message.   
2. If the customer removes an item, the system updates the Shopping Cart and recalculates the total price.   
3. If the system fails to update the Shopping Cart, it logs an error and displays a message to the customer.   
4. If the customer is a guest and modifies the cart, the system updates the session-based cart and prompts the customer to log in to save changes permanently.  
  
Use Case Name: Shopping Cart Deletion   
Use Case ID: UC-08   
Actors: Customer, System   
Preconditions:   
1. The customer is logged into the system or has a session-based Shopping Cart as a guest.   
2. The Shopping Cart contains at least one item.   
3. The system is operational and accessible.   
  
Postconditions:   
1. All items in the Shopping Cart are successfully removed.   
2. The Shopping Cart data entity is cleared.   
3. The system provides a confirmation of cart deletion.   
4. The customer is redirected to a relevant page (e.g., homepage or product catalog).   
  
Main Flow:   
1. The customer navigates to the Shopping Cart page.   
2. The system displays the current items in the cart.   
3. The customer selects the "Delete Cart" or "Clear Cart" option.   
4. The system confirms the deletion request with the customer (e.g., via a pop-up dialog).   
5. If the customer confirms, the system removes all items from the Shopping Cart data entity.   
6. The system updates the cart summary to reflect the empty state.   
7. The system displays a confirmation message (e.g., "Cart has been cleared").   
8. The customer is redirected to a relevant page (e.g., homepage).   
  
Alternative Flow:   
1. If the customer cancels the deletion request, the system retains the cart contents and does not make any changes.   
2. If the system fails to delete the cart items, it logs an error and displays a message to the customer.   
3. If the customer is not logged in and deletes the cart, the session-based cart is cleared and no persistent data is saved.   
4. If the system encounters an issue during redirection, it displays a message confirming the cart deletion and provides a link to the homepage.  
  
Use Case Name: Order Placement   
Use Case ID: UC-09   
Actors: Customer, System, Payment, Inventory, Administrator, Product, Shopping Cart   
  
Preconditions:   
1. The customer is logged into the system or is a guest with a session-based Shopping Cart.   
2. The Shopping Cart contains at least one item.   
3. The system is operational and accessible.   
4. The Inventory data entity has sufficient stock for the items in the cart.   
5. The Payment gateway is available and operational.   
  
Postconditions:   
1. The order is successfully placed and stored in the Order data entity.   
2. The Inventory data entity is updated to reflect the reduction in stock.   
3. A confirmation message is sent to the customer.   
4. The Shopping Cart is cleared or marked as inactive.   
5. The Administrator is notified of the new order for processing.   
  
Main Flow:   
1. The customer accesses the Shopping Cart page and selects "Proceed to Checkout."   
2. The system displays the cart summary, including item details, quantities, and total price.   
3. The customer enters or selects shipping details and confirms the order.   
4. The system validates the shipping information.   
5. The customer selects a payment method and proceeds with the payment via the Payment data entity.   
6. The system processes the payment and confirms its success.   
7. The system creates a new Order record in the Order data entity, including the items, pricing, and shipping information.   
8. The system updates the Inventory data entity to reduce stock levels for the ordered items.   
9. The system sends a confirmation message (e.g., via email) to the customer.   
10. The Shopping Cart is cleared or marked as inactive.   
11. The system logs the order and notifies the Administrator.   
  
Alternative Flow:   
1. If the customer enters invalid shipping information, the system displays an error and prompts for correction.   
2. If the payment fails, the system informs the customer and allows them to retry the payment or choose an alternate method.   
3. If the Inventory is insufficient for any item, the system displays a message and suggests alternatives or allows the customer to update the cart.   
4. If the system fails to create the order, it logs an error and displays a message to the customer, offering to retry the process.   
5. If the customer chooses to cancel the order during checkout, the system returns to the Shopping Cart page without making any changes.   
6. If the system cannot send the confirmation message, it logs an error and provides an option for the customer to request a resend.  
  
Use Case Name: Order Confirmation   
Use Case ID: UC-10   
Actors: Customer, System, Order, Payment, Inventory, Administrator, Contact Information   
  
Preconditions:   
1. The customer has placed an order via the Order Placement use case (UC-09).   
2. The system has successfully created an order record in the Order data entity.   
3. The Payment data entity has confirmed the transaction.   
4. The Inventory data entity has been updated to reflect the order.   
5. The system is operational and accessible.   
  
Postconditions:   
1. The customer receives an order confirmation message.   
2. The order details are finalized and stored in the Order data entity.   
3. The system updates the Inventory data entity to reflect the new stock levels.   
4. The Administrator is notified of the confirmed order.   
5. The Payment status is marked as confirmed.   
6. The Contact Information is associated with the order.   
  
Main Flow:   
1. The system generates an order confirmation message based on the Order data entity.   
2. The system sends the confirmation message (e.g., via email or SMS) to the customer using their Contact Information.   
3. The system updates the order status to "Confirmed" in the Order data entity.   
4. The system logs the order confirmation for audit and tracking purposes.   
5. The Administrator receives a notification of the new confirmed order.   
6. The customer receives a summary of the order, including items, pricing, and shipping details.   
7. The system ensures that the Inventory data entity is updated to reduce the stock of the ordered items.   
8. The system confirms the payment status in the Payment data entity.   
  
Alternative Flow:   
1. If the system fails to send the confirmation message, it logs an error and provides an option for the customer to request a resend.   
2. If the order status cannot be updated, the system logs an error and notifies the customer and Administrator.   
3. If the Inventory update fails, the system rolls back the order and informs the customer.   
4. If the Payment status is not confirmed, the system alerts the customer and Administrator, and may cancel the order.   
5. If the customer’s Contact Information is incomplete or invalid, the system prompts for correction before sending the confirmation.  
  
Use Case Name: Order History Viewing   
Use Case ID: UC-11   
Actors: Customer, System, Order, Contact Information   
  
Preconditions:   
1. The customer is logged into the system.   
2. The system is operational and accessible.   
3. The customer has at least one order stored in the Order data entity.   
  
Postconditions:   
1. The customer can view their order history.   
2. The system displays a summary of previous orders, including order status, total amount, and date.   
3. The customer can access detailed information for each order.   
4. The system logs the order history viewing activity for analytics purposes.   
  
Main Flow:   
1. The customer navigates to the "Order History" section on their account page.   
2. The system retrieves the customer’s order records from the Order data entity.   
3. The system displays a list of past orders, including order ID, date, total amount, and status.   
4. The customer selects a specific order to view more details.   
5. The system displays the detailed information of the selected order, such as items, shipping address, and payment method.   
6. The system provides options for the customer to view receipts or contact support if needed.   
  
Alternative Flow:   
1. If the customer has no previous orders, the system displays a message indicating that their order history is empty.   
2. If the system fails to retrieve the order records, it logs an error and displays a message to the customer.   
3. If the customer’s Contact Information is missing, the system prompts the customer to update their information before viewing receipts or contacting support.   
4. If the order details are not available (e.g., due to a system issue), the system displays an error and offers to retry loading the information.  
  
Use Case Name: Inventory Management   
Use Case ID: UC-12   
Actors: Administrator, System, Inventory, Product, Product Category, Plugin   
Preconditions:   
1. The Administrator is logged into the system.   
2. The system is operational and accessible.   
3. The Inventory data entity is populated with product stock information.   
4. The Product and Product Category data entities are available and correctly configured.   
5. The Plugin is active and supports inventory-related operations.   
  
Postconditions:   
1. The Inventory data entity is updated with the latest stock levels.   
2. The system provides real-time inventory status to users.   
3. Inventory alerts or notifications are generated if stock levels fall below a threshold.   
4. The Administrator receives a confirmation of the inventory update or adjustment.   
5. The system logs the inventory management activity for audit purposes.   
  
Main Flow:   
1. The Administrator navigates to the "Inventory Management" section in the system.   
2. The system retrieves and displays the current inventory levels from the Inventory data entity.   
3. The Administrator selects a product from the Product data entity to update its stock.   
4. The system displays the product details, including category and current stock level.   
5. The Administrator modifies the stock level (e.g., increases or decreases the quantity).   
6. The system validates the new stock level to ensure it is a non-negative number.   
7. The system updates the Inventory data entity with the new stock level.   
8. The system checks if the stock level is below the predefined threshold.   
9. If the stock is below the threshold, the system triggers an alert via the Plugin for restocking.   
10. The system provides a confirmation message to the Administrator that the inventory has been updated.   
11. The system logs the change in the Inventory data entity for audit and tracking.   
  
Alternative Flow:   
1. If the Administrator selects a product that does not exist in the Inventory, the system displays an error and suggests creating a new inventory record.   
2. If the new stock level is negative, the system displays an error and prompts the Administrator to enter a valid quantity.   
3. If the system is unable to update the Inventory, it logs an error and displays a message to the Administrator.   
4. If the Plugin is not active, the system does not trigger restocking alerts but logs the event for review.   
5. If the Administrator cancels the update before submission, the system retains the original inventory data and does not make any changes.   
6. If the system encounters an unexpected error during inventory validation or update, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Product Category Creation   
Use Case ID: UC-13   
Actors: Administrator, System, Product Category, Product   
Preconditions:   
1. The Administrator is logged into the system.   
2. The system is operational and accessible.   
3. The Product data entity is populated with products that need to be categorized.   
4. The Plugin is active and supports category creation functionality.   
  
Postconditions:   
1. A new product category is successfully created in the Product Category data entity.   
2. The system confirms the creation of the category.   
3. The category is associated with relevant products in the Product data entity.   
4. The system logs the category creation activity for audit purposes.   
  
Main Flow:   
1. The Administrator navigates to the "Product Category Management" section of the system.   
2. The system displays a form for creating a new product category.   
3. The Administrator enters the category name, description, and other relevant details.   
4. The system validates the category name for uniqueness and correctness.   
5. The system saves the new category to the Product Category data entity.   
6. The system provides a confirmation message that the category was successfully created.   
7. The Administrator can assign products to the new category or leave it for later use.   
  
Alternative Flow:   
1. If the category name is already in use, the system displays an error message and prompts the Administrator to choose a different name.   
2. If the Administrator enters invalid or incomplete information, the system highlights the missing or incorrect fields and asks for correction.   
3. If the system fails to save the category, it logs an error and displays a message to the Administrator.   
4. If the Plugin is not active, the system still saves the category but does not trigger any integration events.   
5. If the Administrator cancels the operation before submission, the system does not save any changes and returns to the previous page.   
6. If the system encounters an unexpected error during category validation or creation, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Product Category Update   
Use Case ID: UC-14   
Actors: Administrator, System, Product Category, Product, Plugin   
Preconditions:   
1. The Administrator is logged into the system.   
2. The system is operational and accessible.   
3. The target product category exists in the Product Category data entity.   
4. The Plugin is active and supports category update functionality.   
  
Postconditions:   
1. The product category is successfully updated in the Product Category data entity.   
2. The system confirms the category update to the Administrator.   
3. Products associated with the category are updated accordingly.   
4. The system logs the category update activity for audit purposes.   
5. If applicable, the Plugin triggers related events (e.g., notification, synchronization).   
  
Main Flow:   
1. The Administrator navigates to the "Product Category Management" section.   
2. The system displays a list of existing product categories.   
3. The Administrator selects the category to be updated.   
4. The system loads the category details into an editable form.   
5. The Administrator modifies the category name, description, or other attributes.   
6. The system validates the updated information (e.g., uniqueness of category name).   
7. The system saves the updated category to the Product Category data entity.   
8. The system updates associated products in the Product data entity, if necessary.   
9. The system provides a confirmation message that the category was successfully updated.   
10. The system logs the update action for audit and tracking.   
  
Alternative Flow:   
1. If the new category name is already in use, the system displays an error and prompts the Administrator to choose a unique name.   
2. If the Administrator enters invalid or incomplete information, the system highlights the errors and asks for correction.   
3. If the system fails to update the Product Category, it logs an error and displays a message to the Administrator.   
4. If no products are associated with the category, the system only updates the category information without modifying any product data.   
5. If the Plugin is not active, the system still saves the category update but does not trigger any external events.   
6. If the Administrator cancels the update before submission, the system retains the original category data and does not make any changes.   
7. If the system encounters an unexpected error during the update process, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Product Category Deletion   
Use Case ID: UC-15   
Actors: Administrator, System, Product Category, Product, Plugin   
  
Preconditions:   
1. The Administrator is logged into the system.   
2. The system is operational and accessible.   
3. The target product category exists in the Product Category data entity.   
4. The Plugin is active and supports category deletion functionality.   
5. No products are currently assigned to the category, or the Administrator has confirmed the removal of associated products.   
  
Postconditions:   
1. The selected product category is successfully deleted from the Product Category data entity.   
2. The system confirms the deletion to the Administrator.   
3. All associations between the deleted category and products are removed.   
4. If applicable, the Plugin triggers related events (e.g., notification, synchronization).   
5. The system logs the category deletion activity for audit purposes.   
  
Main Flow:   
1. The Administrator navigates to the "Product Category Management" section.   
2. The system displays a list of existing product categories.   
3. The Administrator selects the category to be deleted.   
4. The system checks if the category has any associated products.   
5. If the category has associated products, the system prompts the Administrator to either reassign or remove them before deletion.   
6. The Administrator confirms the deletion request.   
7. The system removes the category from the Product Category data entity.   
8. The system updates the Product data entity to remove the category association from all relevant products.   
9. The system provides a confirmation message that the category was successfully deleted.   
10. The system logs the deletion action for audit and tracking.   
  
Alternative Flow:   
1. If the category does not exist, the system displays an error message and suggests verifying the category name.   
2. If the Administrator attempts to delete a category with associated products but chooses not to proceed with reassignment or removal, the system cancels the deletion.   
3. If the system fails to delete the category, it logs an error and displays a message to the Administrator.   
4. If the Plugin is not active, the system still deletes the category but does not trigger any external events.   
5. If the Administrator cancels the deletion before submission, the system retains the category and does not make any changes.   
6. If the system encounters an unexpected error during the deletion process, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Payment Processing   
Use Case ID: UC-16   
Actors: Customer, System, Payment, Order, Inventory, Administrator   
  
Preconditions:   
1. The customer has selected items in the Shopping Cart and initiated the checkout process.   
2. The system has validated the customer's shipping information.   
3. The Payment gateway is available and operational.   
4. The Inventory data entity has sufficient stock for the items in the cart.   
5. The system is in a secure environment to handle payment information.   
  
Postconditions:   
1. The payment is successfully processed and stored in the Payment data entity.   
2. The order status is updated to "Paid" in the Order data entity.   
3. The Inventory is updated to reflect the reduction in stock for the purchased items.   
4. The customer receives a payment confirmation message.   
5. The Administrator is notified of the completed transaction for monitoring and record-keeping.   
6. The system logs the payment processing activity for audit and analytics.   
  
Main Flow:   
1. The customer selects a payment method (e.g., credit card, PayPal, bank transfer) on the checkout page.   
2. The system collects the necessary payment information (e.g., card number, expiration date, CVV) securely.   
3. The system transmits the payment data to the Payment data entity for processing.   
4. The Payment data entity verifies the transaction with the respective payment gateway.   
5. If the payment is successful, the system updates the Order data entity to reflect the paid status.   
6. The system reduces the stock levels in the Inventory data entity for the purchased items.   
7. The system sends a payment confirmation message (e.g., via email) to the customer.   
8. The system logs the transaction and notifies the Administrator of the completed payment.   
  
Alternative Flow:   
1. If the payment method is not supported or unavailable, the system displays an error and suggests alternative payment options.   
2. If the payment information is invalid, the system rejects the transaction and prompts the customer to re-enter valid details.   
3. If the payment gateway is unreachable, the system logs an error and informs the customer, allowing them to retry later.   
4. If the Inventory is insufficient for any item after payment is processed, the system rolls back the payment and informs the customer.   
5. If the system fails to update the Order or Inventory data entities, it logs an error and displays a message to the customer and Administrator.   
6. If the customer cancels the payment during the process, the system returns to the Shopping Cart page without making any changes.   
7. If the system cannot send the confirmation message, it logs the error and provides an option for the customer to request a resend.  
  
Use Case Name: Payment Information Update   
Use Case ID: UC-17   
Actors: Customer, System, Payment, Contact Information   
  
Preconditions:   
1. The customer is logged into the system.   
2. The system is operational and accessible.   
3. The customer has at least one payment method stored in the Payment data entity.   
4. The Contact Information data entity contains valid customer contact details.   
  
Postconditions:   
1. The customer's selected payment information is updated in the Payment data entity.   
2. The system confirms the successful update of the payment method.   
3. The updated payment information is available for future transactions.   
4. The system logs the update activity for audit purposes.   
  
Main Flow:   
1. The customer navigates to the "Payment Methods" section in their account settings.   
2. The system retrieves and displays the customer's existing payment methods from the Payment data entity.   
3. The customer selects a payment method to update or chooses to add a new one.   
4. The customer enters or modifies the payment details (e.g., card number, expiration date, billing address).   
5. The system validates the payment information for accuracy and completeness.   
6. The system updates the Payment data entity with the new or modified information.   
7. The system provides a confirmation message (e.g., "Payment information updated successfully").   
8. The system logs the update for audit and tracking purposes.   
  
Alternative Flow:   
1. If the customer enters invalid or incomplete payment information, the system displays an error message and prompts for correction.   
2. If the payment information already exists (e.g., duplicate card details), the system informs the customer and suggests using the existing method.   
3. If the system fails to update the Payment data entity, it logs an error and displays a message to the customer.   
4. If the customer’s Contact Information is outdated, the system prompts them to update it to ensure accurate billing or communication.   
5. If the customer cancels the update before submission, the system retains the original payment information and does not make any changes.   
6. If the system encounters an unexpected error during the update process, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Contact Information Management   
Use Case ID: UC-18   
Actors: Customer, System, Contact Information   
  
Preconditions:   
1. The customer is logged into the system.   
2. The system is operational and accessible.   
3. The Contact Information data entity exists and contains the customer's current contact details.   
  
Postconditions:   
1. The customer's contact information is successfully updated or added in the Contact Information data entity.   
2. The system confirms the update or addition of contact information.   
3. The updated contact information is available for use in other use cases (e.g., Order Confirmation, Payment Processing).   
4. The system logs the contact information management activity for audit purposes.   
  
Main Flow:   
1. The customer navigates to the "Contact Information" section in their account settings.   
2. The system retrieves and displays the customer's current contact details from the Contact Information data entity.   
3. The customer modifies existing contact information (e.g., address, phone number, email) or adds a new contact method.   
4. The system validates the updated or new information (e.g., email format, phone number length).   
5. The system saves the changes to the Contact Information data entity.   
6. The system provides a confirmation message (e.g., "Contact information updated successfully").   
7. The system logs the update activity for audit and tracking.   
  
Alternative Flow:   
1. If the customer enters invalid contact information (e.g., incorrect email format), the system highlights the error and prompts for correction.   
2. If the system fails to retrieve the customer's current contact information, it logs an error and displays a message to the customer.   
3. If the system fails to save the updated information, it logs an error and displays a message to the customer.   
4. If the customer cancels the operation before submission, the system retains the original contact information and does not make any changes.   
5. If the system encounters an unexpected error during validation or saving, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Plugin Installation   
Use Case ID: UC-19   
Actors: Administrator, System, Plugin   
  
Preconditions:   
1. The Administrator is logged into the system.   
2. The system is operational and accessible.   
3. The Plugin is available for installation.   
4. The system has the necessary permissions to install and activate the Plugin.   
  
Postconditions:   
1. The Plugin is successfully installed and activated in the system.   
2. The Plugin functionality is available for use by the Administrator or other users.   
3. The system logs the Plugin installation activity for audit and tracking.   
4. The Plugin is integrated with relevant data entities (e.g., Inventory, Order, Product Category).   
  
Main Flow:   
1. The Administrator navigates to the "Plugin Management" section of the system.   
2. The system displays a list of available plugins, including details such as name, version, and functionality.   
3. The Administrator selects the Plugin to be installed and clicks the "Install" button.   
4. The system checks system compatibility and verifies the Plugin's requirements.   
5. The system downloads and installs the Plugin.   
6. The system activates the Plugin and initializes its configuration.   
7. The system confirms the Plugin installation and displays a success message.   
8. The system logs the installation event, including timestamp and Administrator details.   
  
Alternative Flow:   
1. If the Plugin is not compatible with the system, the system displays an error and suggests alternative plugins or updates.   
2. If the Plugin installation fails due to missing dependencies, the system logs an error and displays a message to the Administrator.   
3. If the Plugin is already installed, the system displays a message and suggests updating it instead.   
4. If the system lacks the required permissions to install the Plugin, the system prompts the Administrator to grant the necessary access.   
5. If the Plugin fails to activate after installation, the system logs the event and provides a retry option or troubleshooting guidance.   
6. If the Administrator cancels the installation before completion, the system stops the process and does not save any changes.   
7. If the system encounters an unexpected error during installation, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Plugin Configuration   
Use Case ID: UC-20   
Actors: Administrator, System, Plugin   
  
Preconditions:   
1. The Administrator is logged into the system.   
2. The Plugin is already installed and available for configuration.   
3. The system is operational and accessible.   
4. The Plugin provides configurable settings or parameters.   
  
Postconditions:   
1. The Plugin is configured with the specified settings.   
2. The system applies the configuration changes to the Plugin.   
3. The Plugin functionality is updated according to the new configuration.   
4. The system logs the Plugin configuration activity for audit and tracking.   
  
Main Flow:   
1. The Administrator navigates to the "Plugin Configuration" section in the system.   
2. The system displays the list of installed plugins and their current configuration status.   
3. The Administrator selects a Plugin to configure and opens the configuration interface.   
4. The Administrator modifies the Plugin settings (e.g., API keys, integration options, activation status).   
5. The system validates the configuration inputs for correctness and completeness.   
6. The system saves the updated configuration to the Plugin data entity.   
7. The system applies the changes and activates the new settings in the Plugin.   
8. The system confirms the configuration update and displays a success message.   
9. The system logs the Plugin configuration activity for audit and tracking purposes.   
  
Alternative Flow:   
1. If the Plugin does not support configuration, the system displays a message and does not allow changes.   
2. If the Administrator enters invalid configuration data, the system highlights the errors and prompts for correction.   
3. If the system fails to save the configuration, it logs an error and displays a message to the Administrator.   
4. If the Plugin fails to apply the new configuration, the system displays a warning and provides a retry or troubleshooting option.   
5. If the Administrator cancels the configuration process, the system reverts to the previous configuration and does not apply any changes.   
6. If the system encounters an unexpected error during configuration, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Plugin Uninstallation   
Use Case ID: UC-21   
Actors: Administrator, System, Plugin   
Preconditions:   
1. The Administrator is logged into the system.   
2. The system is operational and accessible.   
3. The Plugin to be uninstalled is currently installed and active.   
4. No critical processes depend on the Plugin at the moment of uninstallation.   
  
Postconditions:   
1. The Plugin is successfully uninstalled from the system.   
2. The Plugin functionality is no longer available to users.   
3. The system logs the Plugin uninstallation activity for audit purposes.   
4. Any Plugin-specific configurations or data are removed or archived.   
  
Main Flow:   
1. The Administrator navigates to the "Plugin Management" section of the system.   
2. The system displays a list of installed plugins, including their status and functionality.   
3. The Administrator selects the Plugin to be uninstalled and clicks the "Uninstall" button.   
4. The system checks if the Plugin is in use or has dependencies.   
5. If the Plugin can be safely uninstalled, the system proceeds with deactivation and removal.   
6. The system removes the Plugin files and deactivates it in the Plugin data entity.   
7. The system confirms the Plugin uninstallation and displays a success message.   
8. The system logs the uninstallation event with details such as timestamp and Administrator.   
  
Alternative Flow:   
1. If the Plugin is currently in use (e.g., managing Inventory or Order notifications), the system displays a warning and prompts the Administrator to resolve dependencies first.   
2. If the Plugin is not installed, the system displays an error message and suggests verifying the Plugin name.   
3. If the system fails to uninstall the Plugin due to file access issues or errors, it logs the event and displays a message to the Administrator.   
4. If the Administrator cancels the uninstallation process, the system retains the Plugin and does not make any changes.   
5. If the system encounters an unexpected error during uninstallation, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Manage Order Confirmation   
Use Case ID: UC-22   
Actors: Administrator, System, Order Confirmation, Order, Customer, Contact Information   
  
Preconditions:   
1. The Administrator is logged into the system.   
2. The system is operational and accessible.   
3. The Order Confirmation data entity contains existing or pending confirmation records.   
4. The Order data entity has at least one order requiring confirmation management.   
5. The Contact Information data entity has valid details for the associated customer.   
  
Postconditions:   
1. The Order Confirmation data entity is updated with the latest status or modifications.   
2. The Order data entity reflects the updated confirmation status.   
3. The system provides the Administrator with confirmation of the changes made.   
4. If applicable, the system sends an updated confirmation message to the customer.   
5. The system logs the Order Confirmation management activity for audit purposes.   
  
Main Flow:   
1. The Administrator navigates to the "Order Confirmation Management" section in the system.   
2. The system retrieves and displays a list of order confirmations from the Order Confirmation data entity.   
3. The Administrator selects a specific order confirmation to review or modify.   
4. The system loads the details of the selected order confirmation, including order ID, customer information, and confirmation status.   
5. The Administrator updates the confirmation status (e.g., from "Pending" to "Confirmed" or "Cancelled").   
6. The system validates the update to ensure it aligns with the order's current state and business rules.   
7. The system updates the Order Confirmation data entity with the new status.   
8. The system synchronizes the change with the Order data entity to reflect the updated confirmation status.   
9. If the new status requires customer notification, the system sends an updated confirmation message via the customer's Contact Information.   
10. The system provides a confirmation message to the Administrator that the order confirmation was successfully updated.   
11. The system logs the activity for audit and tracking.   
  
Alternative Flow:   
1. If the selected order confirmation does not exist, the system displays an error and prompts the Administrator to verify the order ID.   
2. If the Administrator attempts to modify an order confirmation to an invalid status, the system rejects the change and displays an error message.   
3. If the system fails to update the Order Confirmation data entity, it logs an error and displays a message to the Administrator.   
4. If the system fails to send the updated confirmation message to the customer, it logs the error and provides an option to manually resend the message.   
5. If the Administrator cancels the modification before submission, the system retains the original confirmation status and does not make any changes.   
6. If the system encounters an unexpected error during the process, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Manage Cart Item   
Use Case ID: UC-23   
Actors: Customer, System, ShoppingCart, Product, Inventory, Plugin   
  
Preconditions:   
1. The customer is logged into the system or is a guest with an active session-based Shopping Cart.   
2. The Shopping Cart contains at least one item (CartItem) to be managed.   
3. The system is operational and accessible.   
4. The Product and Inventory data entities are correctly populated and available.   
5. If applicable, the Plugin is active and supports cart item management functionality.   
  
Postconditions:   
1. The CartItem is successfully modified, removed, or updated in the ShoppingCart data entity.   
2. The system recalculates the cart total and updates the summary accordingly.   
3. The Inventory data entity reflects any changes due to item updates (e.g., stock adjustments).   
4. The system confirms the changes to the customer or logs them for audit purposes.   
5. If applicable, the Plugin triggers related events (e.g., notifications, analytics tracking).   
  
Main Flow:   
1. The customer navigates to the Shopping Cart page.   
2. The system displays the list of CartItems in the cart, including product name, quantity, price, and total.   
3. The customer selects a specific CartItem for modification (e.g., changing quantity, removing the item).   
4. The system updates the CartItem in the ShoppingCart data entity based on the customer's action.   
5. If the quantity is modified, the system checks the updated quantity against the available stock in the Inventory data entity.   
6. The system recalculates the cart total and updates the summary (e.g., total number of items, total price).   
7. The system provides a confirmation message (e.g., "Your cart has been updated").   
8. The system logs the modification or removal of the CartItem for audit and tracking.   
9. If the Plugin is active, it may trigger events such as analytics tracking for cart modifications.   
  
Alternative Flow:   
1. If the updated quantity exceeds the available stock, the system adjusts the quantity to the maximum available and displays a warning message.   
2. If the customer attempts to modify or remove a CartItem that does not exist in the cart, the system displays an error and prompts the customer to check the item.   
3. If the system fails to update the CartItem in the ShoppingCart data entity, it logs an error and displays a message to the customer.   
4. If the system cannot access the Inventory data entity to verify stock levels, it logs an error and may display a generic message to the customer.   
5. If the customer is a guest and modifies the cart, the system updates the session-based cart and may prompt the customer to log in to save the changes permanently.   
6. If the Plugin is not active, the system still processes the cart item changes but does not trigger any external events.   
7. If the customer cancels the modification or removal before submission, the system retains the original CartItem and does not make any changes.   
8. If the system encounters an unexpected error during cart item management, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Manage Plugin Configuration   
Use Case ID: UC-24   
Actors: Administrator, System, PluginConfiguration, Plugin   
  
Preconditions:   
1. The Administrator is logged into the system.   
2. The system is operational and accessible.   
3. The Plugin is already installed and available for configuration.   
4. The PluginConfiguration data entity exists and contains current configuration settings for the Plugin.   
  
Postconditions:   
1. The PluginConfiguration data entity is updated with the new or modified settings.   
2. The Plugin is reconfigured to reflect the updated settings.   
3. The system confirms the success of the configuration update.   
4. The system logs the PluginConfiguration management activity for audit and tracking purposes.   
5. If applicable, the Plugin triggers events or notifications based on the updated configuration.   
  
Main Flow:   
1. The Administrator navigates to the "Plugin Configuration Management" section in the system.   
2. The system retrieves and displays the current configuration settings of the Plugin from the PluginConfiguration data entity.   
3. The Administrator modifies the configuration parameters (e.g., API keys, feature toggles, timeout settings).   
4. The system validates the new configuration values to ensure they meet the Plugin's requirements.   
5. The system updates the PluginConfiguration data entity with the modified settings.   
6. The system applies the updated configuration to the Plugin.   
7. The Plugin reloads its configuration and adjusts its behavior accordingly.   
8. The system provides a confirmation message (e.g., "Plugin configuration updated successfully").   
9. The system logs the configuration update activity, including the Administrator and timestamp.   
10. If the Plugin is integrated with other data entities (e.g., Inventory, Order), the system ensures the changes are reflected in related processes.   
  
Alternative Flow:   
1. If the Administrator enters invalid or unsupported configuration values, the system displays an error message and prompts for correction.   
2. If the Plugin does not support the configuration being modified, the system displays a warning and prevents the change.   
3. If the system fails to update the PluginConfiguration data entity, it logs an error and displays a message to the Administrator.   
4. If the Plugin fails to apply the updated configuration, the system displays a warning and provides a retry option or troubleshooting guidance.   
5. If the Administrator cancels the configuration process before submission, the system reverts to the previous configuration and does not make any changes.   
6. If the system encounters an unexpected error during configuration validation or application, it displays a generic error message and offers to retry the operation.   
7. If the Plugin is not currently installed, the system displays an error and suggests installing the Plugin before configuration.  
  
Use Case Name: Manage Product   
Use Case ID: UC-25   
Actors: Administrator, System, Product, Product Category, Inventory, Plugin   
  
Preconditions:   
1. The Administrator is logged into the system.   
2. The system is operational and accessible.   
3. The Product data entity is populated with existing product records.   
4. The Product Category and Inventory data entities are available and correctly configured.   
5. If applicable, the Plugin is active and supports product management functionality.   
  
Postconditions:   
1. The Product data entity is updated with the new or modified product information.   
2. The system confirms the success of the product management action (e.g., addition, modification, or deletion).   
3. The product's category and inventory status are updated accordingly.   
4. The system logs the product management activity for audit and tracking purposes.   
5. If applicable, the Plugin triggers related events (e.g., notification, synchronization).   
  
Main Flow:   
1. The Administrator navigates to the "Product Management" section of the system.   
2. The system displays a list of existing products, including their names, categories, prices, and stock levels.   
3. The Administrator selects an option to add, modify, or delete a product.   
4. If adding a new product, the Administrator enters product details such as name, description, price, category, and inventory quantity.   
5. If modifying an existing product, the system loads the product details into an editable form. The Administrator updates the relevant information.   
6. If deleting a product, the system checks if the product is associated with any orders or shopping carts. If not, the Administrator confirms the deletion.   
7. The system validates the product information for correctness, completeness, and adherence to business rules (e.g., non-negative price and inventory levels).   
8. The system updates the Product data entity with the new or modified product information.   
9. If the product is added or modified, the system updates the Product Category and Inventory data entities accordingly.   
10. The system confirms the product management action (e.g., "Product updated successfully," "Product deleted successfully").   
11. The system logs the action in the audit log for tracking purposes.   
12. If the Plugin is active, it may trigger events such as notifications or synchronize with external systems (e.g., third-party marketplaces).   
  
Alternative Flow:   
1. If the Administrator attempts to add a product with a duplicate name or SKU in the Product data entity, the system displays an error and prompts for a unique identifier.   
2. If the product price or inventory quantity is invalid (e.g., negative value), the system rejects the update and displays an error message.   
3. If the selected product category does not exist, the system displays an error and suggests creating or selecting a valid category.   
4. If the Administrator attempts to delete a product that is currently in an active order or shopping cart, the system warns them and prevents deletion until the product is no longer referenced.   
5. If the system fails to update the Product, Product Category, or Inventory data entities, it logs an error and displays a message to the Administrator.   
6. If the Plugin is not active or does not support the requested product management action, the system proceeds with the update but does not trigger external events.   
7. If the Administrator cancels the operation before submission, the system retains the original product information and does not make any changes.   
8. If the system encounters an unexpected error during product validation or update, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Manage Order   
Use Case ID: UC-26   
Actors: Administrator, System, Order, Customer, Contact Information, Inventory   
  
Preconditions:   
1. The Administrator is logged into the system.   
2. The system is operational and accessible.   
3. The Order data entity contains an existing order that needs to be modified or processed.   
4. The Customer associated with the order has valid Contact Information.   
5. The Inventory data entity is accessible and reflects current stock levels if inventory adjustments are needed.   
  
Postconditions:   
1. The Order data entity is updated with the new or modified order status or details.   
2. The Customer is notified of the changes made to the order.   
3. The Inventory data entity is updated if the order is canceled or modified.   
4. The system logs the Manage Order activity for audit and tracking purposes.   
5. The Administrator receives a confirmation of the order modification or processing.   
  
Main Flow:   
1. The Administrator navigates to the "Order Management" section of the system.   
2. The system retrieves and displays a list of existing orders, including order status, customer details, and items.   
3. The Administrator selects a specific order to manage (e.g., to cancel, modify, or update the status).   
4. The system loads the order details into a management interface, displaying information such as order ID, customer name, items, total amount, and current status.   
5. The Administrator modifies the order (e.g., updates status to "Shipped," "Processing," or "Cancelled," or adjusts order details like item quantity).   
6. The system validates the modification to ensure it is allowed based on the current order status and business rules.   
7. If the modification involves changing the item quantity or canceling the order, the system updates the Inventory data entity to adjust stock levels accordingly.   
8. The system updates the Order data entity with the new status or modified details.   
9. The system sends a notification (e.g., via email or SMS) to the Customer using their Contact Information to inform them of the changes.   
10. The system logs the activity, including the Administrator who made the change and the timestamp.   
11. The system confirms the successful management of the order to the Administrator.   
  
Alternative Flow:   
1. If the Administrator attempts to modify an order that is already in a terminal state (e.g., "Cancelled" or "Completed"), the system displays an error and suggests that the order cannot be further modified.   
2. If the system cannot access the Inventory data entity to update stock levels, it logs an error and displays a message to the Administrator.   
3. If the Contact Information for the associated Customer is invalid or incomplete, the system prompts the Administrator to update the Contact Information before sending notifications.   
4. If the system fails to update the Order data entity, it logs an error and displays a message to the Administrator.   
5. If the Administrator cancels the operation before submission, the system retains the original order data and does not make any changes.   
6. If the system encounters an unexpected error during order management, it displays a generic error message and offers to retry the operation.   
7. If the modification requires a refund (e.g., order cancellation), the system initiates a payment reversal via the Payment data entity and confirms the refund status.   
8. If the Inventory update fails due to a stock shortage after modification, the system rolls back the order changes and alerts the Administrator.  
  
Use Case Name: Manage Inventory   
Use Case ID: UC-27   
Actors: Administrator, System, Inventory, Product, Plugin   
  
Preconditions:   
1. The Administrator is logged into the system.   
2. The system is operational and accessible.   
3. The Inventory data entity contains the current stock levels of products.   
4. The Product data entity is populated with relevant product records.   
5. If applicable, the Plugin is active and supports inventory management functionality.   
  
Postconditions:   
1. The Inventory data entity is updated with new or modified stock levels.   
2. The system confirms the success of the inventory management action (e.g., addition, modification, or adjustment).   
3. The Product data entity reflects the updated inventory status for the relevant products.   
4. The system logs the inventory management activity for audit and tracking purposes.   
5. If applicable, the Plugin triggers related events (e.g., restocking alerts, inventory synchronization).   
  
Main Flow:   
1. The Administrator navigates to the "Inventory Management" section of the system.   
2. The system retrieves and displays the current inventory levels for all products from the Inventory data entity.   
3. The Administrator selects a product to update its inventory level.   
4. The system loads the selected product's details, including its current stock level.   
5. The Administrator modifies the inventory level (e.g., increases or decreases stock).   
6. The system validates the new inventory level to ensure it is a non-negative number and within acceptable limits.   
7. The system updates the Inventory data entity with the new stock level for the selected product.   
8. The system synchronizes the updated stock level with the corresponding product in the Product data entity.   
9. The system checks if the updated stock level meets any predefined thresholds for alerts.   
10. If the stock level falls below a threshold, the system triggers an alert via the Plugin for restocking.   
11. The system confirms the successful update of the inventory to the Administrator.   
12. The system logs the inventory management activity, including the Administrator and timestamp.   
  
Alternative Flow:   
1. If the Administrator attempts to modify the inventory level of a product that does not exist in the Inventory data entity, the system displays an error and suggests verifying the product details.   
2. If the new inventory level is invalid (e.g., negative value or exceeds maximum stock limit), the system displays an error and prompts the Administrator to enter a valid quantity.   
3. If the system fails to update the Inventory data entity due to technical issues, it logs an error and displays a message to the Administrator.   
4. If the Plugin is not active, the system still processes the inventory update but does not trigger any external alerts or notifications.   
5. If the Administrator cancels the operation before submission, the system retains the original inventory level and does not make any changes.   
6. If the system encounters an unexpected error during inventory validation or update, it displays a generic error message and offers to retry the operation.   
7. If the system is unable to synchronize the updated inventory level with the Product data entity, it logs an error and displays a message to the Administrator.   
8. If the Administrator requests a bulk update of inventory levels, the system processes the changes and confirms the update for each item, logging the activity accordingly.  
  
Use Case Name: Manage Administrator   
Use Case ID: UC-28   
Actors: System, Super Administrator   
  
Preconditions:   
1. The Super Administrator is logged into the system with elevated privileges.   
2. The system is operational and accessible.   
3. The Administrator data entity is populated with existing administrator records.   
4. The system has the necessary permissions to add, modify, or remove administrators.   
  
Postconditions:   
1. The Administrator data entity is updated with new or modified administrator information.   
2. The system confirms the success of the administrator management action (e.g., creation, modification, or deactivation).   
3. The system logs the administrator management activity for audit and tracking purposes.   
4. If applicable, the Plugin triggers related events (e.g., access control updates, notifications).   
5. The changes are reflected in the system's access control and permissions structure.   
  
Main Flow:   
1. The Super Administrator navigates to the "Administrator Management" section of the system.   
2. The system retrieves and displays a list of existing administrators, including their usernames, roles, and statuses.   
3. The Super Administrator selects an option to add, modify, or deactivate an administrator.   
4. If adding a new administrator, the Super Administrator enters the administrator's details such as username, password, role, and access level.   
5. If modifying an existing administrator, the system loads the administrator's information into an editable form. The Super Administrator updates the relevant fields.   
6. If deactivating an administrator, the system displays a confirmation dialog. The Super Administrator confirms the deactivation.   
7. The system validates the administrator data for correctness, uniqueness (e.g., username), and compliance with security policies.   
8. The system updates the Administrator data entity with the new or modified information.   
9. The system updates the access control and permissions settings based on the changes made.   
10. The system confirms the administrator management action (e.g., "Administrator added successfully," "Administrator role updated," "Administrator deactivated").   
11. The system logs the activity, including the Super Administrator who made the change and the timestamp.   
12. If the Plugin is active, it may trigger events such as sending a notification to the affected administrator or logging the change in external systems.   
  
Alternative Flow:   
1. If the Super Administrator attempts to add an administrator with a duplicate username, the system displays an error and prompts for a unique identifier.   
2. If the entered password does not meet the system's strength requirements, the system rejects the action and asks for a stronger password.   
3. If the Super Administrator attempts to modify or deactivate the last active Super Administrator, the system displays a warning and prevents the change to ensure system integrity.   
4. If the system fails to update the Administrator data entity due to technical issues, it logs an error and displays a message to the Super Administrator.   
5. If the Plugin is not active, the system proceeds with the update but does not trigger any external notifications or events.   
6. If the Super Administrator cancels the operation before submission, the system retains the original administrator information and does not make any changes.   
7. If the system encounters an unexpected error during validation or update, it displays a generic error message and offers to retry the operation.   
8. If the administrator is deactivated, the system ensures that no active sessions or permissions for that administrator are retained in the system.  
  
Use Case Name: Manage Payment   
Use Case ID: UC-29   
Actors: Administrator, System, Payment, Customer, Order, Contact Information   
  
Preconditions:   
1. The Administrator is logged into the system with appropriate permissions to manage payment records.   
2. The system is operational and accessible.   
3. The Payment data entity contains existing payment records that need to be reviewed, modified, or processed.   
4. The Order data entity has associated orders with the payment records.   
5. The Contact Information data entity has valid details for the customers involved in the payments.   
  
Postconditions:   
1. The Payment data entity is updated with new or modified payment records or statuses.   
2. The Order data entity reflects the updated payment status (e.g., "Paid," "Failed," "Refunded").   
3. The system confirms the success of the payment management action to the Administrator.   
4. The system logs the payment management activity for audit and tracking purposes.   
5. If applicable, the customer is notified via their Contact Information of any changes to the payment status.   
6. If a refund is initiated, the Payment and Order data entities are updated accordingly.   
  
Main Flow:   
1. The Administrator navigates to the "Payment Management" section in the system.   
2. The system retrieves and displays a list of existing payments, including payment ID, associated order ID, payment method, amount, and current status.   
3. The Administrator selects a specific payment to review or modify.   
4. The system loads the selected payment's details, such as transaction date, customer information, and payment status.   
5. The Administrator updates the payment status (e.g., from "Pending" to "Paid," "Failed," or "Refunded") or modifies other relevant payment information (e.g., payment method, amount).   
6. The system validates the update to ensure it aligns with the Order status and business rules.   
7. The system updates the Payment data entity with the new or modified information.   
8. The system synchronizes the change with the Order data entity to reflect the updated payment status.   
9. If the payment is updated to "Refunded," the system initiates a refund process via the Payment data entity and updates the Order status accordingly.   
10. The system sends a notification (e.g., via email or SMS) to the customer using their Contact Information to inform them of the changes.   
11. The system confirms the successful management of the payment to the Administrator.   
12. The system logs the activity, including the Administrator who made the change and the timestamp.   
  
Alternative Flow:   
1. If the selected payment does not exist in the Payment data entity, the system displays an error and prompts the Administrator to verify the payment ID.   
2. If the Administrator attempts to modify the payment status in a way that conflicts with the associated Order status (e.g., marking a payment as "Paid" for a canceled order), the system rejects the change and displays an error message.   
3. If the system fails to update the Payment data entity, it logs an error and displays a message to the Administrator.   
4. If the system fails to send a notification to the customer, it logs the error and provides an option to manually retry the notification.   
5. If the Administrator cancels the operation before submission, the system retains the original payment data and does not make any changes.   
6. If the system encounters an unexpected error during the payment management process, it displays a generic error message and offers to retry the operation.   
7. If a refund is initiated but the payment method does not support refunds, the system logs an error and displays a message to the Administrator, suggesting alternative actions.   
8. If the Contact Information for the customer is invalid or incomplete, the system prompts the Administrator to update it before sending notifications.  
  
Use Case Name: Manage Contact Information   
Use Case ID: UC-30   
Actors: Customer, System, Contact Information   
  
Preconditions:   
1. The customer is logged into the system.   
2. The system is operational and accessible.   
3. The Contact Information data entity exists and contains the customer's current contact details.   
  
Postconditions:   
1. The Contact Information data entity is updated with new or modified contact details.   
2. The system confirms the successful update or addition of contact information.   
3. The updated contact information is available for use in other system processes (e.g., order confirmation, payment processing).   
4. The system logs the contact information management activity for audit and tracking purposes.   
  
Main Flow:   
1. The customer navigates to the "Contact Information Management" section in their account settings.   
2. The system retrieves and displays the customer's existing contact information from the Contact Information data entity.   
3. The customer selects an option to add a new contact method or modify an existing one.   
4. The customer enters or updates the contact details (e.g., address, phone number, secondary email).   
5. The system validates the entered information to ensure it meets format and completeness requirements.   
6. The system updates the Contact Information data entity with the new or modified data.   
7. The system provides a confirmation message (e.g., "Contact information updated successfully").   
8. The system logs the activity, including the timestamp and the customer who initiated the change.   
9. The system ensures that the updated information is available for use in related use cases (e.g., Order Confirmation, Payment Processing).   
  
Alternative Flow:   
1. If the customer enters invalid contact information (e.g., incorrect phone number format), the system highlights the error and prompts for correction.   
2. If the system fails to retrieve the customer's current contact information, it logs an error and displays a message to the customer.   
3. If the system fails to update the Contact Information data entity, it logs an error and displays a message to the customer.   
4. If the customer attempts to delete a contact method that is required for system functionality (e.g., primary email or phone number), the system displays a warning and prevents the deletion.   
5. If the customer cancels the operation before submission, the system retains the original contact information and does not make any changes.   
6. If the system encounters an unexpected error during validation or update, it displays a generic error message and offers to retry the operation.  
  
Use Case Name: Manage Plugin   
Use Case ID: UC-31   
Actors: Administrator, System, Plugin, PluginConfiguration   
  
Preconditions:   
1. The Administrator is logged into the system with the necessary permissions to manage plugins.   
2. The system is operational and accessible.   
3. The Plugin data entity contains installed plugins.   
4. The PluginConfiguration data entity is available and can store or update plugin settings.   
5. The system supports plugin management functionalities (installation, activation, deactivation, and removal).   
  
Postconditions:   
1. The Plugin data entity is updated to reflect the current state of the plugin (e.g., active, inactive, removed).   
2. If the plugin is activated or deactivated, its functionality is toggled accordingly.   
3. If the plugin is reconfigured, the PluginConfiguration data entity is updated.   
4. The system logs the plugin management activity for audit and tracking purposes.   
5. The Administrator receives confirmation of the plugin management action.   
6. If the Plugin is removed, any dependent configurations are also deleted or archived.   
  
Main Flow:   
1. The Administrator navigates to the "Plugin Management" section in the system.   
2. The system displays a list of all installed plugins, along with their current status (e.g., active, inactive).   
3. The Administrator selects a plugin to manage, choosing from options such as activate, deactivate, or remove.   
4. If activating or deactivating, the system updates the plugin's status in the Plugin data entity.   
5. The system applies the new status to the plugin, enabling or disabling its functionality.   
6. If removing the plugin, the system checks for dependencies or active configurations.   
7. If no conflicts are found, the system proceeds to remove the plugin and any associated configurations.   
8. The system confirms the plugin management action (e.g., "Plugin activated," "Plugin removed").   
9. The system logs the activity, including the timestamp and the Administrator who performed the action.   
10. If the PluginConfiguration was modified, the system ensures the changes are applied or rolled back as appropriate.   
  
Alternative Flow:   
1. If the Administrator attempts to activate a plugin that is already active, the system displays a message and does not perform any action.   
2. If the plugin is in use (e.g., managing inventory or customer notifications), the system displays a warning before deactivation or removal.   
3. If the system detects that a plugin is critical to system operation, deactivation is prevented or requires additional confirmation.   
4. If the plugin removal fails due to locked files or dependencies, the system logs the error and displays a message to the Administrator.   
5. If the system fails to update the plugin status or configuration, it logs an error and provides a message to the Administrator.   
6. If the Administrator cancels the operation before submission, the system reverts to the previous state and does not make any changes.   
7. If the system encounters an unexpected error during plugin management, it displays a generic error message and offers to retry the operation.   
8. If the PluginConfiguration is modified during the process but the plugin fails to apply the changes, the system displays a warning and provides troubleshooting options.  
  
Use Case Name: Manage Order Confirmation   
Use Case ID: UC-32   
Actors: Administrator, System, Order, Customer, Order Confirmation, Contact Information   
  
Preconditions:   
1. The Administrator is logged into the system.   
2. The system is operational and accessible.   
3. The Order data entity contains an order that has been placed and is pending confirmation.   
4. The Order Confirmation data entity is available and supports the management of confirmation records.   
5. The Contact Information data entity has valid details for the associated customer.   
  
Postconditions:   
1. The Order Confirmation data entity is updated with the new confirmation status or details.   
2. The Order data entity reflects the updated confirmation status.   
3. The system confirms the successful management of the order confirmation to the Administrator.   
4. The system logs the activity for audit and tracking purposes.   
5. If required, the system sends a confirmation or update message to the customer via their Contact Information.   
  
Main Flow:   
1. The Administrator navigates to the "Order Confirmation Management" section of the system.   
2. The system retrieves and displays a list of orders that are pending confirmation.   
3. The Administrator selects an order for which the confirmation status needs to be updated.   
4. The system loads the order details, including the customer information, product list, and current confirmation status.   
5. The Administrator modifies the confirmation status (e.g., from "Pending" to "Confirmed," "Cancelled," or "On Hold").   
6. The system validates the new confirmation status to ensure it aligns with business rules and order state.   
7. The system updates the Order Confirmation data entity with the new status.   
8. The system synchronizes the change with the Order data entity to reflect the updated confirmation status.   
9. If the updated status requires customer notification, the system sends a message (e.g., email or SMS) using the customer's Contact Information.   
10. The system logs the activity, including the Administrator who performed the action and the timestamp.   
11. The system provides a confirmation message to the Administrator indicating the successful management of the order confirmation.   
  
Alternative Flow:   
1. If the selected order is not in a "Pending" state, the system displays a message and does not allow the confirmation status to be changed.   
2. If the system fails to update the Order Confirmation data entity, it logs an error and displays a message to the Administrator.   
3. If the system fails to send the confirmation message to the customer, it logs the error and provides an option for the Administrator to manually retry the notification.   
4. If the Administrator cancels the operation before submission, the system retains the original confirmation status and does not make any changes.   
5. If the system encounters an unexpected error during the process, it displays a generic error message and offers to retry the operation.   
6. If the Contact Information for the customer is missing or invalid, the system prompts the Administrator to update or correct it before sending any notifications.   
7. If the Order data entity cannot be updated due to system constraints, the system logs the error and prevents the confirmation status from being changed.